

CS 241: Computer Organization and Assembly Language

Midterm

Do not open until instructed to do so.

Name: _____

“C’est en forgeant qu’on devient forgeron.”

(“It is by smithing that one learns to become a smith.”)

~French proverb

Every problem is marked with a ► . When you see this symbol, it means that’s a question which you can — and should — answer.

For grader use:

Score: _____

Syscalls

0	sys_read
1	sys_write
60	sys_exit

Arguments in: rdi, rsi, rdx, r10, r8, r9

Return value in: rax

Callee-saved regs.: rcx, r11

C-style functions

```
func:
    push rbp
    mov rbp, rsp

    ...

    pop rbp
    ret
```

Arguments in: rdi, rsi, rdx, rcx, r8, r9

Return value in: rax

Callee-saved regs.: rbx, rbp, r12-r15

Caller-saved regs.: rax, r10, r11, arguments

Memory operands

size [*displacement* + *base* + *m* * *offset*]

size byte, word, dword, etc.

displacement Constant address of array

base Array base register

m 1, 2, 4, or 8

offset Array offset register

Instructions

mov rm, rmi	Move
xchng rm, rm	Swap
lea r, m	Load Effective Address
xor r, r	Set <i>r</i> to 0

add rm, rmi	Addition
sub rm, rmi	Subtraction
mul rmi	Unsigned multiply (by/into rax)
div rm	Unsigned divide (by/into rax, rdx)
imul rmi	Signed multiply
idiv rmi	Signed divide

cmp rm, rmi	Compare (subtract), update flags
test rm, rmi	Test, update flags

jmp target	Jump to target
jcc target	Jump if condition <i>CC</i>
loop target	Decrement rcx, jump if not 0

call func	Push rip, jump to func
ret	Pop rip and jump to it
push rmi	Push onto stack
pop rmi	Pop from stack

r: register, m: memory operand, i: immediate

Condition codes

<i>CC</i>	Meaning
-----------	---------

a	Unsigned >
ae	Unsigned ≥
b	Unsigned <
be	Unsigned ≤
g	Signed >
ge	Signed ≥
l	Signed <
le	Signed ≤
e	=
ne	≠
s, c, z, ...	If flag is set

5 points each

- Perform the following binary addition:

01110101 + 00111111

Show your work (all carries).

$$\begin{array}{r}
 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1 \\
 +\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1 \\
 \hline
 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0
 \end{array}$$

- What is the decimal value of 10001111 when interpreted as two's complement-signed?

$$10001111b = -113$$

- What is the two's complement binary value of -17?

$$-17 = 11101111b$$

- Suppose a cache has a hit percentage of 98%. The latency for a hit is 1ns, while the latency for a miss is 300ns. What is the average latency of a memory access?

$$\begin{aligned}
 \text{Avg. latency} &= 0.98(1\text{ns}) + (1 - 0.98) * 300\text{ns} \\
 &= 0.98 + 6 \\
 &= 6.98
 \end{aligned}$$

- Suppose a cache has a total size of 16kB and a line size of 256 bytes. How many sets, and how many lines/set does the cache have if it is:

	Set-associative?	Fully-associative?
Sets:	64	1
Lines/Set:	1	64

- Perform the addition 11100010 + 01101111, show your work, write the final sum, as well as the state of the flags after the addition is complete.

$$\begin{array}{r}
 1\ 1\ 1\ 1\ 1 \\
 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0 \\
 +\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1 \\
 \hline
 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1
 \end{array}$$

$$CF = 1$$

$$OF = 0$$

$$SF = 0$$

$$ZF = 0$$

► For each of the following condition codes, write the state of the flags it will check:

- a ZF == 0 and CF == 0
- b CF == 1
- s SF == 1
- ne ZF == 0

► When executing a `syscall`, which of the following is used for the **syscall code**, the **1st argument**, the **2nd argument**, and the **3rd argument**?

- rdx 3rd
- rax syscall code
- rsi 2nd
- rdi 1st

► What registers are used when passing floating-point arguments to a C-ABI-compatible function?

xmm0 - xmm7

► Write assembly code to perform the division 157 / 13 using the `div` instruction. When your code is complete, the quotient should be in `rax`.

```
mov rdx, 0      ; or xor rdx, rdx
mov rax, 157
mov rbx, 13
div rbx
```

25 points each

► Complete the following `syscall`-style function so that it will print out a rectangle made of `#` characters. E.g., if the function's parameter in `rdi` is 5, it should print out

```
#####
#####
#####
#####
#####
```

Do not modify the `.data` section.

```
section .data                                     jne .inner_loop

newline:    db    10                               ; Print newline
star:       db    '*'                             mov rax, 1
                                                    mov rdi, 1
                                                    mov rsi, newline
                                                    mov rdx, 1
                                                    syscall

section .text                                     ; Decrement and repeat outer loop
print_stars:                                     dec r12
    ; Size in rdi                               cmp r12, 0
                                                    jne .outer_loop

    ; The problem doesn't say if the size can be 0.
    cmp rdi, 0
    je .end

    mov r12, rdi    ; Outer loop index
    mov r14, rdi    ; Saved size
                                                    .end:
                                                    ret

.outer_loop:

    mov r13, r14    ; Inner loop index
.inner_loop:

    ; Print one #
    mov rax, 1
    mov rdi, 1
    mov rsi, star
    mov rdx, 1
    syscall

    ; Decrement and repeat inner loop
    dec r13
    cmp r13, 0
```

- Complete the following function so that it returns 1 if the (qword) value in rdi is found within the array pointed to by rsi, with length (in bytes) in rdx.

```
section .text
```

```
contains:
```

```
    ; rdi = search target  
    ; rsi = addr. of array  
    ; rdx = length of array (bytes)  
    ; Return 1 in rax if found, 0 if not
```

```
    add rdx, rsi ; End of the array
```

```
    mov rax, 0
```

```
.loop:
```

```
    cmp rsi, rdx  
    je .return
```

```
    cmp qword [rsi], rdi  
    jne .no_match
```

```
    ; Found!  
    mov rax, 1  
    jmp .return
```

```
.no_match:
```

```
    add rsi, 8 ; Move forward 8 bytes  
    jmp .loop
```

```
.return
```

```
    ret
```